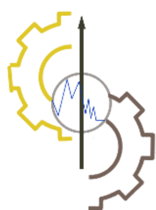
E-ISSN: 2717-4581 | P-ISSN: 2783-1345



Innovation Management and Operational Strategies

www.journal-imos.ir

Inov. Manage. Oper. Stra. Vol. 3, No. 4 (2022) 452–464.





Paper Type: Original Article

New Heuristic Local Search Method for University Course Timetabling Problem

Mohammad Sadegh Shiri[1,*], Seyed Mostafa Khorramizadeh[2], Vahid Ahmadi[2]

[1] Department of Applied Mathematics, Faculty of Basic Sciences, Islamic Azad University of Arsanjan Branch, Arsanjan, Iran; MS.Shiri@iau.ac.ir.

[2] Department of Optimization, Faculty of Mathematical Sciences, Shiraz University of Technology, Shiraz, Iran; m.khorrami@sutech.ac.ir; v.ahmad@sutech.ac.ir.

Citation:



Shiri, M. S., Khorramizadeh, M., & Ahmadi, V. (2022). New heuristiv local search method for university course timetabling. *Innovation management and operational strategies*, 3(4), 452-464.

Received: 01/03/2022 Reviewed: 04/04/2022 Revised: 22/04/2022 Accepted: 01/06/2022

Abstract

Purpose: This paper presents a new two-phase method for solving the curriculum-based university course timetabling problem. A new metaheuristic approach is used in both phases of the new present method.

Methodology: A feasible, high-quality solution is computed in the first phase of the new method. To this end, the hard constraints relating to the periods are considered, and a solution is computed that satisfies these hard constraints. In the next step, a new method is introduced for assigning rooms to courses, after which a feasible solution is calculated based on the solution that satisfies the period's hard constraints. In the second phase, several new neighbourhood functions are used to improve the quality of computed feasible solutions. While the fitness function of the first phase is based on the violation of hard constraints, the fitness function of the second phase is based on the penalty of the feasible solution.

Findings: The numerical results indicate that the required computing time increases with the size of instances, and the algorithm tends to converge towards the optimal solution after a few minutes.

Originality/Value: The presented algorithm enables us to deal with extensive university course timetabling problems in practice. Moreover, it provides us with an efficient way to obtain feasible solutions to such real-world instances and try to improve their quality.

Keywords: Course timetabling, Heuristic method, Local search, Scheduling, Tabu search.

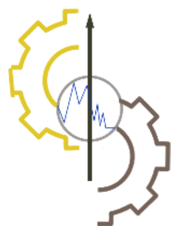
https://doi.org/10.22105/imos.2022.332030.1215

Corresponding Author: MS.Shiri@iau.ac.ir



http://dorl.net/dor/20.1001.1.27831345.1401.3.4.6.5





Innovation Management & Operational Strategies

مدیریت نوآوری و راهبردهای عملیاتی

دوره ۳، شماره (۴)، (۱۴۰۱)، ۴۶۴-۴۵۲
www.journal-imos.ir





نوع مقاله: پژوهشی

روش جستجوی محلی ابتکاری جدید برای جدول زمانی دروس دانشگاهی

محمدصادق شیری[۱،*]، سید مصطفی خرمی زاده[۲]، وحید احمدی[۲]

[۱]گروه ریاضی کاربردی، دانشکده علوم پایه و مهندسی، دانشگاه آزاد اسلامی واحد ارسنجان، ارسنجان، ایران.

[۲]گروه بهینه‌سازی، دانشکده ریاضی، دانشگاه صنعتی شیراز، شیراز، ایران.

چکیده

هدف: در این مقاله یک روش دومرحله‌ای جدید برای حل مساله‌ی زمان‌بندی دروس دانشگاهی مبتنی بر برنامه‌ی درسی ارایه شده است. در هر دو مرحله، روش از رویکرد فراابتکاری جدید استفاده شده است. علاوه بر این، یک نمایش جواب جدید برای زمان‌بندی دروس دانشگاهی معرفی‌شده است و از برخی رویکردها نیز برای تشدید و تنوع استفاده می‌شود که کاملا مبتنی بر نمایش جواب جدید است.

روش‌شناسی پژوهش: در مرحله‌ی اول روش جدید، یک جواب باکیفیت بالا قابل‌اجرا محاسبه می شود. برای این منظور، ابتدا محدودیت های سخت مربوط به دوره‌های زمانی در نظر گرفته‌شده و جوابی محاسبه می‌شود که این محدودیت‌های سخت را برآورده کند. در مرحله‌ی بعد روش جدیدی برای تخصیص اتاق‌ها به دروس معرفی می‌شود که پس از اعمال آن بر روی جوابی که محدودیت‌های سخت دوره‌ی زمانی را برآورده می‌کند، یک جواب شدنی محاسبه می‌شود. علاوه بر این، نتایج عددی نشان می‌دهد که جواب شدنی محاسبه‌شده کیفیت بالایی دارد. در مرحله‌ی دوم، ابتدا چندین تابع همسایگی جدید برای بهبود کیفیت جواب شدنی محاسبه‌شده به‌طور قابل‌توجهی مورداستفاده قرار می‌گیرد که برای کاهش جریمه جواب شدنی محاسبه‌شده مرحله‌ی اول طراحی شده است. درحالی‌که تابع تناسب مرحله‌ی اول مبتنی بر نقض محدودیت‌های سخت است، تابع تناسب مرحله‌ی دوم بر اساس جریمه‌ی جواب شدنی است. در بسیاری از الگوریتم‌های فراابتکاری که تاکنون ارایه شده‌اند، تلاش محاسباتی زیادی بر روی الگوریتم برای انتساب اتاق‌ها به دوره‌ها صرف می‌شود. ویژگی جدید الگوریتم ارایه شده این است که از یک استراتژی برای تخصیص اتاق‌ها به دوره فقط یک‌بار و بدون استفاده از هیچ الگوریتم تطبیقی استفاده می‌شود.

یافته‌ها: الگوریتم ارایه شده بر روی برخی از نمونه‌های استاندارد ادبیات اعمال‌شده و کارایی الگوریتم ارایه شده مورد تجزیه‌وتحلیل قرار گرفته است. نتایج عددی نشان می‌دهد که زمان محاسبات موردنیاز با اندازه‌ی نمونه‌ها افزایش می‌یابد و الگوریتم بعد از چند دقیقه به سمت جواب بهینه همگرا می‌شود.

اصالت/ارزش افزوده علمی: الگوریتم ارایه شده ما را قادر می‌سازد تا در عمل با مسایل بزرگ زمان‌بندی دروس دانشگاهی مواجه شویم. علاوه بر این، روشی کارآمد برای دستیابی به جواب‌های شدنی برای نمونه‌های دنیای واقعی و تلاش برای بهبود کیفیت آن‌ها در اختیار ما قرار می‌دهد.

کلیدواژه‌ها: جدول زمانی دروس دانشگاهی، جستجوی محلی، جستجوی ممنوعه، روش ابتکاری، زمان‌بندی.

۱- مقدمه

یکی‌از وظایف ضروری دانشگاه‌ها که معمولا چالش‌برانگیز است، مساله‌ی زمان‌بندی دروس دانشگاهی است. حل مساله‌ی زمان‌بندی دروس دانشگاهی یک کار بزرگ و پیچیده است و دانشگاه هرسال و هر ترم با این مساله مواجه بوده و به‌طور مستقیم بر عملکرد آموزشی اثر

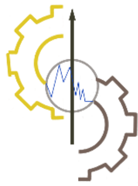
* نویسنده مسئول

MS.Shiri@iau.ac.ir



http://dorl.net/dor/20.1001.1.27831345.1401.3.4.6.5

می‌گذارد. در مساله‌ی زمان‌بندی دروس دانشگاهی، منابع دانشگاه شامل واحدها، اساتید، کلاس‌ها و سایر موارد آموزشی به یک سری از دوره‌های زمانی اختصاص داده می‌شوند. اگر آن‌ها به‌طور مناسب تخصیص داده شوند، کیفیت آموزشی می‌تواند افزایش یابد.



مجموعه‌ای از دروس، مجموعه‌ای از اساتید، مجموعه‌ای از گروه‌ها (دانشجویان شرکت‌کننده در یک درس)، مجموعه‌ای از کلاس‌ها و مجموعه‌ای از دوره‌های زمانی را در نظر بگیرید. مساله‌ی زمان‌بندی دروس دانشگاهی[1] تخصیص دوره‌های زمانی و کلاس‌ها به دروس است به‌گونه‌ای که مقررات آموزشی رعایت شود. دو نوع مهم از این مساله، زمان‌بندی دروس دانشگاهی بر اساس برنامه‌ی آموزشی (برنامه‌ی درسی) و زمان‌بندی دروس دانشگاهی بر اساس ثبت‌نام است. در اینجا، تمرکز بر روی زمان‌بندی دروس دانشگاهی، بر اساس برنامه‌ی آموزشی است. مجموعه دروسی را که با رعایت قوانین و برای گرفتن مدرک تحصیلی نیاز است که موردمطالعه قرار گیرند، برنامه آموزشی نامیده می‌شود. تخصیص کلاس‌ها و دوره‌های زمانی به دروس بر این اساس، جدول زمان‌بندی دروس بر اساس برنامه‌ی آموزشی نامیده می‌شود. در حالتی که زمان‌بندی دروس دانشگاهی بر اساس ثبت‌نام باشد، دروس در جدول زمانی به‌گونه‌ای قرار داده می‌شوند تا همه دانشجویان بتوانند در جلساتی که در آن ثبت‌نام کرده‌اند شرکت کنند (کوالیزا و سرافینی[2]، ۲۰۰۴؛ سشیا و همکاران[3]، ۲۰۲۲). مساله‌ی زمان‌بندی دروس دانشگاهی مبتنی بر برنامه‌ی درسی، شامل تعیین بهترین زمان‌بندی دروس دوره دانشگاهی در یک بازه زمانی معین، اختصاص دروس هر درس به کلاس‌ها و دوره‌های زمانی است، به‌طوری‌که یک سری محدودیت‌ها برآورده شود. محدودیت‌های *UCTP* به محدودیت‌های سخت و نرم تقسیم می‌شوند. هر جوابی که یک محدودیت سخت را نقض کند قابل‌قبول نیست. تخصیصی که تمام محدودیت‌های سخت را برآورده کند، جواب شدنی نامیده می‌شود. در *UCTP*، هدف یافتن جوابی شدنی است که نقض محدودیت‌های نرم یا جریمه‌ی مربوط به آن را کمینه کند. الگوریتم ارایه‌شده در این مقاله، تلاش می‌کند تا جریمه‌ی جواب شدنی را با برآورده کردن محدودیت‌های سخت زیر کمینه کند.

۱. هر زوج از (دوره زمانی، کلاس) یک زیرجلسه نامیده می‌شود. باید برای هر درس c در یک هفته n_c زیرجلسه مشخص کنیم. هر درس می‌تواند یک یا دو جلسه در یک هفته داشته باشد. اگر یک درس یک جلسه داشته باشد، آن جلسه دارای n_c زیرجلسه در یک هفته است. اگر یک درس دارای دو جلسه باشد، n_c زوج است و هر جلسه دارای $n_c/2$ زیرجلسه در هفته است.
۲. هر گروه نمی‌تواند در یک دوره‌ی زمانی در بیش از یک درس شرکت کند.
۳. هر استاد نمی‌تواند بیش از یک درس را در یک دوره‌ی زمانی تدریس کند.
۴. دوره‌های زمانی هر جلسه از هر درس باید فشرده باشد. اگر دو دوره‌ی زمانی از هر جلسه درس در یک روز برنامه‌ریزی‌شده باشد، باید به دوره‌های زمانی متوالی اختصاص داده شود.
۵. در هر کلاس نمی‌توان بیش از یک درس در یک دوره‌ی زمانی قرار داد.
۶. تمام ساعات هر جلسه از هر درس برنامه‌ریزی‌شده در یک روز، باید در همان کلاس قرار گیرد.

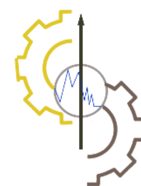
در این مقاله‌ی یک الگوریتم دومرحله‌ای ارایه می‌دهیم. در مرحله‌ی اول الگوریتم، با استفاده از روش جدید جستجوی محلی ابتکاری (*HLS*) و با برآورده کردن محدودیت‌های سخت فوق، یک جواب شدنی را برای مساله‌ی *UCTP* پیداکرده و سپس یک روش کارآمد برای تخصیص کلاس ارایه خواهیم کرد. در مرحله‌ی دوم الگوریتم، با حفظ شدنی بودن جواب و با اختصاص چندین همسایگی جدید و متفاوت از همسایگی‌های مرحله‌ی اول، جریمه‌ی جواب شدنی را کاهش می‌دهیم. یکی دیگر از نوآوری‌های این مقاله معرفی یک نمایش جواب جدید برای مساله‌ی *UCTP* است. توجه کنید که با توجه به جدید بودن نمایش جواب، ساختارهای همسایگی معرفی‌شده در این مقاله نیز کاملا جدید هستند. علاوه بر این، رهیافت‌هایی که به‌منظور تشدید و تنوع استفاده شده‌اند نیز کاملا مخصوص نمایش جواب جدید هستند.

۲- ادبیات و بیشینه‌ی تحقیق

در مقاله‌ی کوالیزا و سرافینی (۲۰۰۴)، نویسندگان طراحی و اجرای یک سیستم مبتنی بر کامپیوتر را برای کمک به ساخت جدول زمانی ترکیبی درس و امتحانات دانشگاهی گزارش کردند. این سیستم از یک مدل برنامه‌نویسی عدد صحیح (*IP*) استفاده می‌کند که دروس را به

---

[1] University Course Timetabling Problem (UCTP)
[2] Qualizza and Serafini
[3] Ceschia et al.



دوره‌ی زمانی و کلاس‌ها اختصاص می‌دهد. در مقاله‌ی سشیا و همکاران (۲۰۲۲)، توسعه‌ای از برنامه‌نویسی منطقی مقید را برای طراحی یک زمان‌بندی خودکار برای دانشگاه پردو به کاربردند که در آن محدودیت‌های نرم با استفاده از وزن‌های شخصی به‌طور جزیی برآورده می‌شوند. سپس نویسندگان مقاله‌ی گاسپرو و شارف[1] (۲۰۰۲)، استفاده از تکنیک‌های جستجوی محلی را بر اساس ترکیب‌های مختلف توابع همسایگی بررسی کردند و آن را برای یک مساله‌ی زمان‌بندی به کار بردند. در مارتین[2] (۲۰۰۴)، از یک مدل برنامه‌ریزی عدد صحیح برای اختصاص اساتید به دروس، کلاس‌های درس و بازه‌های زمانی استفاده شده است. سپس در داسکلاکی و همکاران[3](۲۰۰۴)، یک فرمول برنامه‌نویسی باینری عدد صحیح جدید را برای مساله‌ی جدول زمانی دانشگاه ارایه کردند. این مدل محدودیت‌هایی را برای تعداد زیادی از قوانین عملیاتی و الزامات موجود در اکثر موسسات دانشگاهی فراهم می‌کند. نویسندگان مقاله‌ی آولا و واسیلو[4](۲۰۰۵)، یک مطالعه‌ی موردی موفق را توصیف کردند که در آن با استفاده از الگوریتم شاخه و برش، جواب بهینه‌ی یک مساله‌ی زمان‌بندی واقعی برای دروس دانشگاهی، به دست می‌آید.

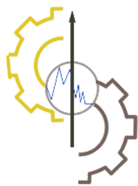
داسکلاکی و بیرباس[5] (۲۰۰۵)، یک روش آزادسازی دومرحله‌ای را ارایه دادند که مساله‌ی جدول زمانی دانشگاه را به‌صورت کارا با استفاده از فرمول برنامه‌ریزی اعداد صحیح حل می‌کند. سپس در مقاله‌ی کوالیزا و سرافینی (۲۰۰۴)، یک رویکرد برنامه‌ریزی خطی عدد صحیح را بر اساس تولید ستون پیشنهاد کردند به‌طوری‌که هر ستون با جدول زمانی هفتگی یک درس مرتبط است. سپس تحقیقی از یک رویکرد فراابتکاری ژنتیک ساده بر روی مجموعه‌ای از ابتکارهای سازنده پرکاربرد (ابتکارهای رنگ‌آمیزی گراف) در جدول زمانی ارایه کردند. در پژوهش شیملپفنگ و هلبر[6] (۲۰۰۷)، یک رویکرد برنامه‌نویسی عدد صحیح (*IP*) توصیف شده است که جدول زمانی کامل همه دروس در دانشکده‌ی اقتصاد و مدیریت در دانشگاه هانوفر آلمان، برای یک ترم با این رویکرد پیاده‌سازی و اجرا شد. در پژوهش سویانتو[7] (۲۰۱۰)، یک الگوریتم ژنتیک هوشمندانه توصیف شده است. الگوریتم ژنتیک با استفاده از مقداردهی اولیه حریصانه و عملگر جهش، دروس عملی دانشگاهی و مساله‌ی جدول زمانی دانشجویی را حل می‌کرد. سپس، نویسندگان مقاله‌ی نگوین و همکاران[8] (۲۰۱۱) کاربرد الگوریتم جستجوی همسایگی متغیر و هفت نوع از آن را بر روی مساله‌ی زمان‌بندی دروس دانشگاهی بر اساس برنامه‌ی آموزشی بسیار محدود در دنیای واقعی ارایه کردند. در چن و شی[9] (۲۰۱۳) دو نوع بهینه‌سازی ازدحام ذرات (*PSO*)، برای زمان‌بندی دروس دانشگاهی ارزیابی شده است و تجزیه بندرز را برای مساله‌ی زمان‌بندی درس بر اساس برنامه‌ی درسی به کار برده‌اند. رویکرد آن‌ها بر اساس تقسیم‌بندی مساله به مسایل زمان‌بندی و تخصیص کلاس بود. سپس از الگوریتم بندرز برای ایجاد برش‌هایی استفاده کردند که برنامه‌ی زمانی و تخصیص کلاس را به هم مرتبط می‌کرد.

لموس و همکاران[10] (۲۰۲۱) به مساله‌ی آشفتگی حداقلی (*MPP*) در زمان‌بندی دروس دانشگاهی می‌پردازند. آن‌ها نشان دادند که با توجه به مجموعه‌ای از اختلالات که باعث می‌شود یک جدول زمانی دیگر شدنی نباشد، جواب *MPP* نزدیک‌ترین جدول زمانی جدید شدنی نسبت به جدول زمانی اولیه است. سپس یک مدل جدید و کامل با انطباق‌های مناسب که برای مطالعه موردی واقعی سال اول دوره ریاضیات دانشگاه کاتانیا، ایتالیا به کار گرفته شده است را تدوین کردند. آن‌ها هم محدودیت‌های برنامه‌ریزی و هم محدودیت‌های مربوط به فشرده بودن برنامه‌های درسی، توزیع دروس (در بازه‌ی زمانی موردبررسی)، اولویت اساتید، حداقل تعداد روزهای کاری، حداکثر ظرفیت و پایداری کلاس‌ها (که باهدف به حداقل رساندن رفت‌وآمدهای روزانه دانشجویان در بین کلاس‌ها انجام می‌شود) را در نظر گرفتند. سشیا و همکاران (۲۰۲۲)، یک الگوریتم فراابتکاری مبتنی بر الگوریتم ژنتیک مرتب‌سازی غیرغالب برای زمان‌بندی دروس دانشگاهی پیشنهاد داده است که زمان‌بندی دوره‌های تحصیلات تکمیلی دانشگاه را تعیین کرده است. شاخص‌های کیفیت آموزشی مانند شایستگی اساتید در تدریس، دوره‌ی مشاوره دانشجویی و مرحله‌ی مناسب برای ارایه آموزش‌ها در طول روز در این مدل لحاظ شده است.

الگوریتم ارایه شده در این مقاله از دو مرحله تشکیل شده است. در مرحله‌ی اول، یک جواب شدنی پیدا می‌کنیم که محدودیت‌های سخت مساله را برآورده کند. برای این مرحله، ابتدا یک روش جدید جستجوی محلی ابتکاری (*HLS*) را برای یافتن جواب مساله ارایه می‌کنیم که محدودیت‌های سخت مربوط به دوره‌های زمانی (محدودیت‌های ۱، ۲، ۳ و ۴) را برآورده می‌کند. روش جدید جستجوی محلی ابتکاری

---

[1] Gaspero and Schaerf
[2] Martin
[3] Daskalaki et al.
[4] Avella and Vasil'Ev
[5] Daskalaki and Birbas.
[6] Schimmelpfeng and Helber
[7] Suyanto
[8] Nguyen et al.
[9] Chen and Shih
[10] Lemos et al.



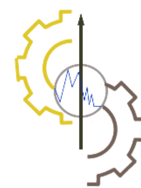
(*HLS*) از ایده‌های الگوریتم فراابتکاری جستجوی ممنوعه استفاده می‌کند. یکی از ویژگی‌های مهم جستجوی محلی ابتکاری این است که با پایان یافتن آن، می‌توانیم کلاس‌هایی را با بازه‌های زمانی مشخص به جلسات دروس اختصاص دهیم تا محدودیت‌های سخت ۵ و ۶ برآورده شوند. سپس یک روش کارآمد برای تخصیص کلاس ارایه خواهیم کرد. در مرحله‌ی دوم الگوریتم ارایه شده، سعی می‌کنیم جریمه‌ی جواب شدنی را کاهش دهیم. برای این کار، چندین همسایگی جدید را به‌صورت تخصصی فراهم می‌کنیم تا جریمه‌ی جواب شدنی را کاهش دهیم و درعین‌حال شدنی بودن جواب را حفظ کنیم. یکی دیگر از ویژگی‌های جستجوی محلی ابتکاری ارایه شده این است که توابع همسایگی مرحله‌ی اول و دوم کاملا متفاوت هستند.

تاکنون در بسیاری از الگوریتم‌های فراابتکاری ارایه شده، از یک الگوریتم تطبیق برای تخصیص کلاس‌ها به دروس و دوره‌های زمانی استفاده شده است. این بخش از الگوریتم نیاز به تلاش محاسباتی زیادی دارد. روش ارایه شده نیازی به استفاده از الگوریتم تطبیق در هر تکرار ندارد. علاوه بر این، تخصیص دوره‌های زمانی به دروس انجام می‌شود تا همیشه تخصیص کلاس‌ها به دروس امکان‌پذیر باشد. این ویژگی، محاسبات روش ارایه شده در این مقاله را به میزان قابل‌توجهی کاهش می‌دهد. یکی دیگر از تفاوت‌های عمده بین الگوریتم ارایه شده با سایر الگوریتم‌های فراابتکاری، فرمول‌بندی مساله است. در مرحله‌ی اول الگوریتم ارایه شده در این مقاله، همراه با سایر محدودیت‌های سخت، فشرده بودن زیرجلسه‌های هر جلسه از دروس برنامه‌ریزی‌شده در همان روز (محدودیت ۴) و سازگاری کلاس‌های زیرجلسه‌های هر جلسه از دروس (محدودیت ۶) به‌عنوان محدودیت‌های سخت را در نظر می‌گیریم. ما نتوانستیم مقاله‌ای در مورد الگوریتم‌های فراابتکاری برای *UCTP* پیدا کنیم که این محدودیت‌ها را به‌عنوان محدودیت‌های سخت در نظر بگیرد. توجه داشته باشید که این ویژگی مرحله‌ی دوم الگوریتم ارایه شده را کاملا متفاوت (و سخت‌تر) از سایر رویکردها می‌کند؛ زیرا با حفظ شدنی بودن جواب، سعی در به حداقل رساندن جریمه دارد. محدودیت‌های ۴ و ۶ در بسیاری از دانشگاه‌های کشورمان محدودیت‌های سختی هستند و در بسیاری از مدل‌های ارایه شده برای *UCTP* به‌عنوان محدودیت‌های نرم در نظر گرفته می‌شوند؛ بنابراین، در نظر گرفتن این محدودیت‌ها به‌عنوان محدودیت‌های سخت، ازنظر مطالعاتی ارزشمند است. همان‌طور که قبلا اشاره کردیم، الگوریتم ارایه شده شامل دو مرحله است. در مرحله‌ی اول یک جواب شدنی پیدا می‌کنیم که محدودیت‌های سخت مساله را برآورده کند. این مرحله در بخش ۲ توضیح داده خواهد شد. در مرحله‌ی دوم الگوریتم، سعی می‌کنیم با حفظ شدنی بودن، جریمه جواب را کاهش دهیم. این مرحله در بخش ۳ موردبحث قرار خواهد گرفت. درنهایت بخش ۴ به نتایج عددی می‌پردازد.

۳- توصیف مرحله‌ی اول

در این بخش، مرحله‌ی اول الگوریتم ارایه شده را شرح می‌دهیم. مرحله‌ی اول از دو زیرمرحله تشکیل شده است. در زیر مرحله‌ی اول یک جواب برای مساله پیدا می‌کنیم که محدودیت‌های سخت مربوط به دوره‌های زمانی مانند محدودیت‌های ۱، ۲، ۳ و ۴ را برآورده می‌کند. برای این زیرمرحله، یک روش جستجوی محلی ابتکاری (*HLS*) ارایه می‌کنیم. در این روش جستجوی محلی ابتکاری، هر جواب با فهرستی از دروس نشان داده می‌شود. درس c با یک ساختار متشکل از دو عدد صحیح و چهار لیست فرعی نشان داده می شود. اولین لیست فرعی از اعضای داده‌ی n_c تشکیل شده است. *ith* عضو داده این لیست فرعی حاوی یک عدد صحیح است که تعیین می‌کند کدام دوره‌ی زمانی به جلسه‌ی *ith* درس c اختصاص داده شده است. در ادامه برای سادگی در نمادگذاری، این لیست فرعی را لیست زمانی درس c می‌نامیم. لیست فرعی دوم نیز از اعضای داده n_c تشکیل شده است. *ith* عضو داده این لیست فرعی حاوی یک عدد صحیح است که تعیین می‌کند کدام کلاس به جلسه‌ی *ith* درس c اختصاص داده شده است. در ادامه برای سادگی در علامت‌گذاری، این لیست فرعی را لیست کلاس درس c می‌نامیم. لیست فرعی سوم اعداد صحیح مربوط به گروه‌های شرکت‌کننده در درس c را ذخیره می‌کند. درنهایت، لیست فرعی چهارم شامل اعداد صحیحی است که کلاس‌های ممکن درس c را مشخص می‌کند، یعنی کلاس‌هایی که می‌توان برای درس c برنامه‌ریزی کرد. دو عدد صحیح از ساختار داده نشان‌دهنده n_c و یک عدد صحیح مربوط به استاد درس c است. اگر یک درس دارای دو جلسه باشد، اولین $n_c = 2$ عضو هر زیر لیست متعلق به بخش اول و آخرین $n_c = 2$ عضو متعلق به لیست فرعی دوم است.

توجه داشته باشید که ازآن‌جایی‌که n_c دوره‌های زمانی را برای درس c مشخص کرده‌ایم، محدودیت ۱ همیشه با نمایش جواب برآورده می شود. توجه داشته باشید که محدودیت‌های ۲، ۳ و ۴ فقط مربوط به تخصیص دوره‌های زمانی است، درحالی‌که محدودیت‌های ۵ و ۶ مربوط به تخصیص کلاس‌ها هستند. در ادامه توضیح می‌دهیم که چگونه می‌توانیم روش جستجوی محلی ابتکاری را طوری طراحی کنیم که در پایان روش، امکان اختصاص کلاس‌ها به دروس وجود داشته باشد تا محدودیت‌های ۵ و ۶ برآورده شوند. در ابتدا مفهوم یک دوره‌ی



زمانی معتبر برای یک درس داده‌شده را تعریف می‌کنیم. فرض کنید برای هر دوره‌ی زمانی t، S_t مجموعه دروسی است که t را به‌عنوان یکی از دوره‌های زمانی خود دارند. در ادامه برای هر درس، منظور از مجموعه کلاس‌های امکان‌پذیر آن درس، مجموعه کلاس‌هایی است که امکان برگزاری آن درس در آن‌ها وجود دارد. می‌دانیم که برخی از دروسی که در یک ساعت برگزار می‌شوند ممکن است برای برگزاری نیاز به کلاس‌های کاملا متفاوتی داشته باشند به‌عبارت‌دیگر، برای هر t ممکن است S_t دارای دو عضو باشد که مجموعه کلاس‌های امکان‌پذیر آن‌ها اشتراک تهی دارند. برای در نظر گرفتن این مفهوم در ادامه، گردایه‌ی زیرمجموعه‌هایی از S_t را که مجموعه‌ی کلاس‌های امکان‌پذیر آن‌ها باهم اشتراک نکته‌ی دارند، با $\{C_{t,k}\}_{k\in I}$ نشان می‌دهیم. برای هر دوره‌ی زمانی t و $k\in I$، فرض کنید $R_{t,k}$ اجتماع مجموعه‌ی کلاس های امکان‌پذیر دروس $C_{t,k}$ است. با توجه به تعریف، برای هر $k\in I$، داریم $R_{t,k}\neq\varnothing$.

تعریف ۱– دوره‌ی زمانی t برای همه‌ی دروس S_t معتبر است، اگر برای هر $k\in I$، تعداد دروس در $C_{t,k}$ کمتر یا مساوی تعداد کلاس‌ها در $R_{t,k}$ باشد، یعنی:

$$\left|C_{t,k}\right|\leq\left|R_{t,k}\right| \quad \forall k\in I. \qquad (1)$$

که در آن $\left|C_{t,k}\right|$ و $\left|R_{t,k}\right|$ به ترتیب نشان‌دهنده تعداد اعضای $C_{t,k}$ و $R_{t,k}$ هستند.

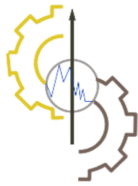
هنگامی‌که اولین زیرمرحله‌ی پایان می‌یابد، جوابی به دست می‌آوریم که محدودیت‌های ۱، ۲، ۳ و ۴ را برآورده می‌کند. قضیه بعدی شرایطی را بیان می‌کند که در آن، برای چنین جوابی می‌توانیم کلاس‌هایی را به دروسی اختصاص دهیم که قیود ۵ و ۶ برآورده شوند.

قضیه ۱– جوابی که قیود سخت ۱، ۲، ۳ و ۴ را برآورده کند در نظر بگیرید. برای این جواب، اگر دوره‌های زمانی به کلاس‌هایی اختصاص داده شود که هر دوره‌ی زمانی t برای همه‌ی دروس در S_t معتبر باشد، می‌توانیم کلاس‌ها را به دروسی اختصاص دهیم که محدودیت‌های ۵ و ۶ برآورده شوند.

اثبات: روش زیر را در نظر بگیرید. برای هر دوره‌ی زمانی t، مجموعه دروس t یعنی S_t را در نظر می‌گیریم. توجه کنید که هر درس در S_t، دقیقا یک زیرجلسه دارد که در دوره‌ی زمانی t برنامه‌ریزی شده است. برای هر درس در S_t ابتدا تعیین می‌کنیم که t متعلق به کدام جلسه از درس است. تمام زیرجلسه‌های آن جلسه را پیمایش می‌کنیم. اگر کلاسی قبلا به یک زیرجلسه از جلسه اختصاص داده شده باشد، همان کلاس را به زیرجلسه‌ی برنامه‌ریزی‌شده آن جلسه اختصاص می‌دهیم. سپس این کلاس را به مجموعه کلاس‌های ممنوعه‌ی t اضافه می‌کنیم. اگر کلاسی به هیچ‌یک از زیرجلسه‌های آن جلسه اختصاص داده نشده باشد، یک کلاس غیرممنوعه دلخواه را در مجموعه کلاس‌های ممکن درس مربوطه انتخاب کرده و آن را به زیرجلسه‌ی برنامه‌ریزی‌شده برای t اختصاص می‌دهیم. در اینجا، توجه داشته باشید که چون تمام دوره‌های زمانی برای دروس مربوطه‌ی خود معتبر است، پس **رابطه (۱)** همیشه معتبر است؛ بنابراین، حداقل یک کلاس ممنوعه در مجموعه کلاس‌های ممکن درس مربوطه وجود دارد که می‌توان آن را به زیرجلسه‌ی برنامه‌ریزی‌شده برای t اختصاص داد. برای یک دوره‌ی زمانی معین t، کلاس اختصاص داده‌شده به مجموعه کلاس‌های ممنوعه t تعلق ندارد؛ بنابراین، محدودیت سخت ۵ با جواب حاصل برآورده می‌شود. ازآنجایی‌که یک کلاس را به تمام زیرجلسه‌های یک درس مشخص اختصاص می‌دهیم، محدودیت سخت ۶ با جواب حاصل برآورده می‌شود.

برای تعیین اینکه آیا یک دوره‌ی زمانی برای درس c معتبر است یا نه لیست شاخص را تعریف می‌کنیم. لیست شاخص به‌گونه‌ای است که هر عضو داده‌ی لیست شاخص، ساختاری متشکل از یک عدد صحیح و دو لیست فرعی دیگر است. عدد صحیح نشان‌دهنده‌ی دوره‌ی زمانی مربوطه t است. اعضای داده‌های لیست فرعی اول، اشاره به دروسی در S_t و اعضای داده‌های لیست دوم شامل یک اشاره‌گر به کلاس که دوره‌ی زمانی متناظر آن برابر با t و یک اشاره‌گر به درس مربوطه است، تشکیل شده‌اند. لیست فرعی اول در طول زیرمرحله‌ی اول برای تعیین اعتبار دوره‌ی زمانی یک درس استفاده می‌شود، درحالی‌که لیست دوم در پایان زیرمرحله‌ی اول برای مرتبط کردن دروس به کلاس‌ها استفاده می‌شود به‌طوری‌که محدودیت‌های ۵ و ۶ برآورده می‌شوند. در ادامه، ابتدا بحث می‌کنیم که چگونه می‌توانیم دوره‌های زمانی را به دروس اختصاص دهیم که محدودیت ۱، ۲، ۳ و ۴ برآورده شده و شرایط قضیه ۱ برقرار باشد. سپس، الگوریتمی را برای تخصیص کلاس ارایه می‌کنیم که توسط اثبات قضیه پیشنهادشده است.

۱-۳- زیرمرحله‌ی اول



در ابتدای الگوریتم یک جواب تصادفی تولید می‌کنیم. برای این منظور، همه زیرجلسه‌های دروس را پیمایش می‌کنیم و برای هر یک دوره‌های زمانی ایجاد می‌کنیم تا وقتی‌که یک زیرجلسه‌ی معتبر پیدا شود. سپس، دوره‌ی زمانی را به آن زیرجلسه اختصاص می‌دهیم و لیست شاخص را بر این اساس به‌روز می‌کنیم. ابتدا باید برای هر جواب مساله یک مقدار برازندگی تعریف کنیم، به‌طوری‌که وقتی مقدار برازندگی یک جواب صفر است، قیود ۲، ۳ و ۴ توسط جواب برآورده شوند. ابتدا محدودیت‌های ۲ و ۳ را در نظر می‌گیریم و اعضای فهرست‌های زمانی دروسی را که در نقض محدودیت‌های ۲ و ۳ نقش دارند ذخیره می‌کنیم. لیست به‌دست‌آمده، لیست نقض اشتراک نامیده می‌شود. مقدار برازندگی اشتراکی برابر با تعداد عناصر لیست نقض اشتراک است. برای به دست آوردن یک همسایه‌ی اشتراکی (N_1) برای جواب فعلی، ابتدا به‌طور تصادفی یکی از اعضای لیست نقض اشتراک را انتخاب می‌کنیم و سپس دوره‌های زمانی تصادفی را ایجاد می‌کنیم تا زمانی که یک دوره‌ی معتبر (که با دوره زمانی فعلی متفاوت است) پیدا شود. اگر مقدار برازندگی اشتراکی همسایه بهتر از مقدار برازندگی اشتراکی جواب فعلی باشد، آنگاه جواب فعلی را با جواب ایجادشده جایگزین می‌کنیم. جوابی را که با اعمال روش فوق به جواب فعلی s به دست می‌آید، همسایه‌ی N_1 می‌نامیم و می‌نویسیم $x \in N_1(s)$.

در مرحله‌ی بعد، در مورد محدودیت ۴ بحث می‌کنیم. تمام جلسه‌های همه دروس را پیمایش می‌کنیم. سپس برای هر جلسه، لیست زمانی مربوطه را پیمایش و روش زیر را اعمال می‌کنیم. بدون از دست دادن کلیت مساله، فرض کنید که اندازه‌ی لیست زمانی n_c است. اعضای اول و دوم لیست زمانی را در نظر می‌گیریم. اگر پشت سر هم نباشند، دومی را به لیست جدیدی اضافه می‌کنیم. در ادامه، این لیست جدید را لیست نقض فشردگی می‌نامیم. سپس اعضای دوم و سوم لیست زمانی را در نظر می‌گیریم. اگر آن‌ها متوالی نباشند، عضو سوم را به لیست نقض فشردگی اضافه می‌کنیم. این روش را برای اعضای سوم و چهارم، چهارم و پنجم و غیره؛ و (n_c-1) امین و n_c امین، تکرار می‌کنیم و اگر دوره زمانی متوالی نباشند، دومین مورد را به لیست نقض فشردگی اضافه می‌کنیم. پس از اعمال روش فوق در تمامی مقاطع کلیه دروس، لیست نقض فشردگی را به دست می‌آوریم. سپس، لیست نقض فشردگی را پیمایش می‌کنیم. برای هر عضو داده‌ی این لیست، می‌دانیم که دوره زمانی متناظر و دوره زمانی قبلی آن (مثلا t) در لیست زمانی درس مربوطه متوالی نیستند؛ بنابراین، برای تعیین اینکه آیا $t+1$ یک دوره‌ی زمانی معتبر برای درس مربوطه است یا نه از لیست شاخص استفاده می‌کنیم. اگر $t+1$ برای درس مربوطه معتبر باشد، آنگاه $t+1$ را به عضوی از لیست نقض فشردگی اختصاص می‌دهیم. در ادامه جوابی را که با اعمال روش فوق بر جواب فعلی s به دست می‌آید، همسایه‌ی N_2 از s می‌نامیم و می‌نویسیم $x \in N_2(s)$. پس از اعمال همسایگی N_2 به جواب فعلی، جواب فعلی را با همسایه‌ی آن جایگزین می‌کنیم و روش فوق را برای هر یک از آن‌ها اعمال می‌کنیم. توجه داشته باشید که اگر تعداد اعضای لیست نقض فشردگی صفر باشد، **محدودیت (۴)** با جواب فعلی برآورده می‌شود. مقدار برازندگی فشردگی برابر با تعداد عناصر لیست نقض فشردگی است. اکنون، ارزش برازندگی یک جواب مساله را تعریف می‌کنیم.

تعریف ۲- مقدار برازندگی یک جواب مساله برابر است با مجموع مقدار برازندگی اشتراکی و مقدار برازندگی فشردگی آن.

۱-۱-۳- روش آشفتگی

حالتی را در نظر بگیرید که در آن مقدار برازندگی اشتراکی صفر است و محدودیت‌های ۲ و ۳ برآورده شده ولی مقدار برازندگی فشردگی صفر نیست. در این حالت اگر همه‌ی اعضای لیست نقض فشردگی را پیمایش کنیم و نتوانیم هیچ‌یک از آن‌ها را تغییر دهیم، روش آشفتگی را به شرح زیر اعمال می‌کنیم. در این مورد، اولین دوره‌های زمانی همه‌ی جلسه‌ها را به یک لیست جدید اضافه می‌کنیم. این لیست را لیست آشفتگی می‌نامیم. در مرحله‌ی بعد، به‌طور تصادفی یک عضو از لیست آشفتگی را انتخاب کرده و دوره‌های زمانی را ایجاد می‌کنیم تا وقتی‌که یک عضو معتبر پیدا شود. سپس، دوره‌ی زمانی معتبر تولیدشده را به عضو انتخابی اختصاص می‌دهیم. در ادامه جواب x را که با اعمال روش فوق در جواب فعلی s به دست می‌آید، همسایه‌ی N_3 از s نامیده و می‌نویسیم $x \in N_3(s)$.

۲-۱-۳- استفاده از ایده‌های جستجوی ممنوعه

در این بخش نشان می‌دهیم که می‌توانیم از ایده‌های روش جستجوی ممنوعه برای بهبود کارایی الگوریتم پیشنهادی استفاده کنیم. برای این کار یک لیست ممنوعه شامل آخرین حرکات l تولید کرده و از انتخاب هر یک از این حرکات ممنوعه توسط الگوریتم پیشنهادی

جلوگیری می‌کنیم. در اینجا مقدار پارامتر l به نمونه‌ی مساله‌ی ورودی بستگی دارد و به‌صورت تجربی تعیین خواهد شد. در الگوریتم پیشنهادی، لیست ممنوعه، فهرستی با اندازه‌ی l است که از آخرین نشانگرهای l به اعضای فهرست‌های زمانی دروسی که برای تولید همسایگان نوع N_l انتخاب کرده‌ایم، تشکیل شده است. همان‌طور که قبلا اشاره کردیم، الگوریتم پیشنهادی، ابتدا به‌صورت تصادفی یک درس را انتخاب می‌کند و سپس به‌صورت تصادفی، عضوی از لیست زمانی درس انتخاب‌شده را در نظر می‌گیرد. حال اگر عضو منتخب لیست زمانی متعلق به لیست ممنوعه باشد، آن را قبول نمی‌کند و این کار را تا زمانی که عضوی انتخاب شود که آن عضو از لیست ممنوعه نباشد، تکرار می‌کنیم. در غیر این صورت، عضو انتخاب‌شده را می‌پذیریم و آن را به اولین لیست ممنوعه اضافه می‌کنیم. اگر پس از اضافه شدن عضو جدید، طول لیست ممنوعه از پارامتر l بیشتر باشد، آخرین عضو لیست ممنوعه را حذف می‌کنیم.

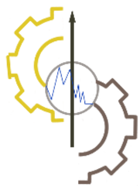
۳-۱-۳- جستجوی محلی ابتکاری

در این بخش، روش جستجوی محلی ابتکاری پیشنهادی (HLS) را شرح می‌دهیم که در الگوریتم ۱ ارایه شده است. در الگوریتم ۱، $C(s)$ مقدار برازندگی فشردگی جواب فعلی s را نشان می‌دهد، $V(s)$ نشان‌دهنده‌ی مقدار برازندگی اشتراکی است. در هر تکرار الگوریتم، ابتدا مقدار برازندگی s را در نظر می‌گیریم. اگر مقدار برازندگی s صفر نباشد، به‌طور تصادفی یک همسایه‌ی $x \in N_1(s)$ ایجاد می‌کنیم. سپس x را ارزیابی کرده و اگر مقدار برازندگی همسایه‌ی تولیدشده کمتر از مقدار برازندگی جواب فعلی باشد، جواب فعلی را باهمسایه‌ی ایجادشده x جایگزین می‌کنیم. در مرحله‌ی بعدی الگوریتم، مقدار برازندگی فشردگی را در نظر گرفته که چنانچه این مقدار صفر نباشد، یک همسایه‌ی $x \in N_2(s)$ ایجاد کرده و اگر بتوانیم جواب را تغییر دهیم، جواب فعلی را با x جایگزین می‌کنیم. اگر مقدار برازندگی فشردگی صفر نباشد اما مقدار برازندگی اشتراکی صفر باشد، روش آشفتگی را با تولید $x \in N_3(s)$ و قرار دادن $s = x$ اعمال می‌کنیم.

در ادامه فرض کنید C مجموعه دروس، R مجموعه‌ی کلاس‌ها، T مجموعه‌ی دوره‌های زمانی و Q مجموعه‌ی زیرجلسات یک دوره‌ی زمانی است. روش جستجوی محلی ابتکاری برای جدول زمانی دروس دانشگاهی:

الگوریتم ۱- یافتن یک جواب شدنی که محدودیت‌های سخت زمانی را برآورده کند.
Algorithm 1- Finding a feasible solution that meets the strict time constraints.

روش جستجوی محلی ابتکاری	
مرحله‌ی ۱:	
$C(s)$ را محاسبه کنید.	▷ مقادیر فشردگی
مرحله‌ی ۲:	
$V(s)$ را محاسبه کنید.	▷ مقادیر برازندگی اشتراکی
$f(s) \leftarrow C(s) + V(s)$	
اگر $f(s) = 0$ آنگاه برگردید.	
اگر $V(s) \neq 0$ آنگاه	
$x \in N_1(s)$ را تولید کنید.	▷ تولید همسایه‌ی اشتراکی
اگر $V(x) \leq V(s)$ پس	
قرار دهید $s = x$.	
اگر $C(s) = 0$، سپس به مرحله‌ی ۲ بروید.	
اگر $V(s) \neq 0$ باشد	
$x \in N_2(s)$ را تولید کنید.	▷ تولید همسایه‌ی فشردگی
قرار دهید $s = x$.	
در غیر این‌صورت	
$x \in N_3(s)$ ایجاد کنید.	▷ اعمال آشفتگی
$s = x$.	
بروید به مرحله‌ی ۱.	



۲-۳- زیرمرحله‌ی دوم

در این بخش، تخصیص کلاس را پس از اعمال روش جستجوی محلی پیشنهادی و به دست آوردن جوابی برای مساله که محدودیت‌های ۱، ۲، ۳ و ۴ را برآورده می‌کند، توضیح می‌دهیم. روش تخصیص کلاس در الگوریتم ۲ توضیح داده شده است. در این الگوریتم $\{1,2,3,...,T\}$ مجموعه‌ای از تمام دوره‌های زمانی را نشان می‌دهد. برای $1 \le t \le T$، $S_t = \{c_1^t, c_2^t, ..., c_{n_t}^t\}$ مجموعه‌ای از دروس t را نشان می‌دهد. $sec(k,t)$ جلسه‌ای از درس c_k^t را نشان می‌دهد که دوره‌ی زمانی t به آن تعلق دارد. $lec(k,t)$ زیرجلسه‌ی $sec(k,t)$ مربوط به دوره‌ی زمانی t است. $FBD(t)$ مجموعه‌ای از کلاس‌های ممنوعه t است. $PR(c_k^t)$ مجموعه‌ای از کلاس‌های ممکن c_k^t است. متغیر z_k^t برابر با r است، اگر کلاس r به حداقل یک زیرجلسه (و درنتیجه همه‌ی زیرجلسه‌ها) از $sec(k,t)$ اختصاص داده شود و در غیر این صورت برابر ۱ است.

می‌دانیم که هر عضو داده‌ی لیست شاخص از یک عدد صحیح و دو لیست فرعی تشکیل شده است. عدد صحیح نشان‌دهنده‌ی یک دوره‌ی زمانی است، مثلا t. اعضای داده‌های اولین لیست فرعی، اشاره‌گر به دوره‌های درسی S_t هستند. برای انجام اختصاص کلاس، برای هر عضو لیست شاخص که عدد صحیح آن‌ها برابر با دوره‌ی زمانی t است، لیست فرعی دوم را پیمایش می‌کنیم. برای هر عضو از لیست فرعی دوم، ابتدا اشاره‌گر کلاس را در نظر می‌گیریم. می‌دانیم که یکی از اعضای لیست زمانی این درس برابر با t است. اگر هر کلاس را به یکی از اعضای لیست کلاس این درس اختصاص دهیم، همان کلاس را به عضو مربوطه در لیست کلاس اختصاص می‌دهیم و این کلاس را به مجموعه کلاس‌های ممنوعه‌ی t اضافه می‌کنیم. در غیر این صورت یک کلاس دلخواه از مجموعه کلاس‌های احتمالی این درس که جزو کلاس‌های ممنوعه نیست را به این عضو لیست کلاس اختصاص می‌دهیم. اگر از این روش برای اختصاص کلاس استفاده کنیم، هر کلاس نمی‌تواند میزبان بیش از یک درس در یک دوره‌ی زمانی باشد و تمام ساعات یک درس برنامه‌ریزی‌شده در یک روز باید در همان کلاس باشد؛ بنابراین، قیود ۵ و ۶ توسط جواب برآورده می‌شوند و یک جواب برای مساله به دست می‌آوریم که تمام محدودیت های مساله را برآورده می‌کند.

الگوریتم ۲- تخصیص کلاس برای جوابی که محدودیت‌های سخت زمانی را برآورده می‌کند.
Algorithm 2- Class assignment for the answer that meets the strict time constraints.

```
t ← 1
مرحله:
اگر t > T آنگاه برگردید.
k ← 1،
اگر k > n_t آنگاه t ← t+1 و به مرحله بروید.
اگر z_k^t = r آنگاه کلاس را به lec(k,t) تخصیص دهید.
اگر z_k^t = -1 آنگاه انتخاب کنید r ∈ PR(c_k^t) − FBD(t) و r را به lec(k,t) اختصاص دهید.
```

در اینجا توجه کنید در روند زیر مرحله‌ی اول و در بدترین حالت، به‌منظور تولید جواب پایه‌ای آغازین، پیمایش همه‌ی زیرجلسات و اختصاص دوره‌ی زمانی تا یافتن زمان معتبر برای هر درس نیاز به $O(|T||Q||C|)$ محاسبات دارد. علاوه بر این، تولید یک همسایه‌ی جواب فعلی s در $N_1(s)$ در بدترین حالت نیاز به $O(|T|^2)$ و تولید یک همسایه‌ی s در $N_2(s)$ در بدترین حالت به‌اندازه‌ی $O(|T||Q|)$ است. با توجه به اینکه تعداد تکرارهای الگوریتم متناهی است می‌توان گفت که پیچیدگی مرحله‌ی اول الگوریتم پیشنهادی $O(|T||Q|(|C|+1)+|T|^2)$ است.

۴- توصیف مرحله‌ی دوم

در مرحله‌ی دوم الگوریتم پیشنهادی، سعی می‌کنیم جریمه‌ی جواب شدنی را کاهش دهیم. برای کاهش جریمه از چهار تابع جدید همسایگی استفاده می‌کنیم. این توابع همسایگی مربوط به مبادله‌ی دوره‌های زمانی دو جلسه انتخاب‌شده، جایگزینی تمام دوره‌های زمانی یک جلسه انتخابی با دوره‌های زمانی مختلف، مبادله‌ی کلاس‌های دو جلسه انتخابی و جایگزینی کلاس‌های یک جلسه انتخاب شده است. مبادله و جایگزینی‌ها با حفظ شدنی بودن جواب حاصل انجام می‌شود. در ادامه توابع همسایگی را توضیح می‌دهیم. ازآنجایی‌که جایگزینی، یک مورد خاص از مبادله است، فقط مبادله‌ی دوره‌های زمانی و کلاس‌های دو جلسه را شرح می‌دهیم.

۴-۱- مبادله‌ی دوره‌های زمانی

روش مبادله دوره‌های زمانی دو جلسه با استفاده از الگوریتم ۳ توضیح داده شده است. در اینجا، $C(s_i)$ درس مربوط به جلسه s_i، $1 \le i \le 2$ را نشان می‌دهد. $T(s_i)$ و $G(s_i)$ نشان‌دهنده‌ی استاد و گروه مربوط به درس $C(s_i)$ هستند. $S_T(s_i)$ مجموعه‌ای از دروس تدریس شده توسط $T(s_i)$ را نشان می‌دهد، $S_G(s_i)$ مجموعه دروسی است که اعضای $G(s_i)$ در آن شرکت می‌کنند $nl(s_i)$ تعداد زیرجلسه‌های جلسه‌ی s_i است، $M_T(s_i)$ مجموعه‌ای از دوره‌های زمانی است که به زیرجلسه‌های دروس در $S_T(s_i)$ اختصاص می‌یابد و $M_G(s_i)$ مجموعه دوره-های زمانی اختصاص داده شده به زیرجلسه‌های دروس در $S_G(s_i)$ را نشان می‌دهد. علاوه بر این، $P_{old}(s_1,s_2)$ و $P_{new}(s_1,s_2)$ جریمه‌ی قبل و بعد از مبادله‌ی دوره‌های زمانی s_1 و s_2 هستند. اگر s_1 و s_2 متعلق به یک درس باشند، متوقف می‌شویم. اگر تعداد جلسات s_1 و s_2 برابر نباشد، متوقف می‌شویم. در غیر این صورت، ابتدا دوره‌های زمانی s_1 را در نظر می‌گیریم. برای هر دوره‌ی زمانی s_1، مثلا t، دروس گروه و استاد مربوط به درس s_2 را پیمایش می‌کنیم. اگر هر یک از این دروس عضو S_t باشد، دوره‌های زمانی این جلسه‌ها قابل‌مبادله نیست، در غیر این صورت بازه‌های زمانی s_2 را در نظر می‌گیریم و روش فوق را تکرار می‌کنیم. اگر متوجه شدیم که دوره‌های زمانی این جلسه‌ها را می‌توان بدون نقض هرگونه محدودیت سخت مبادله کرد، پس از انجام مبادله، جریمه‌ی جواب حاصل را محاسبه می‌کنیم. اگر مبادله جریمه را کاهش دهد، مبادله را انجام داده و ساختار لیست زمانی را به‌روز می‌کنیم، در غیر این صورت مبادله را انجام نمی‌دهیم.

الگوریتم ۳- مبادله‌ی دوره‌های زمانی دو جلسه‌ی s1 و s2.

Algorithm 3- Exchange of time periods of two sessions s1 and s2.

اگر $nl(s_1) \ne nl(s_2)$ یا $C(s_1)=C(s_2)$ آنگاه برگردید.
اگر $M_T(s_1)\cap k(s_1) \ne \varnothing$ یا $M_T(s_2)\cap k(s_2) \ne \varnothing$ آنگاه برگردید.
اگر $M_G(s_1)\cap k(s_1) \ne \varnothing$ یا $M_G(s_2)\cap k(s_2) \ne \varnothing$ آنگاه برگردید.
$P_{old}(s_1,s_2)$ و $P_{new}(s_1,s_2)$ را محاسبه کنید.
اگر $P_{old}(s_1,s_2) < P_{new}(s_1,s_2)$ آنگاه،
دوره‌های زمانی s_1 و s_2 را مبادله کنید.
داده‌های موجود در لیست شاخص را بر این اساس به‌روز کنید.

۴-۲- مبادله‌ی کلاس‌ها

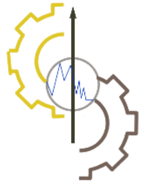
مبادله‌ی کلاس‌های دو جلسه‌ی s_1 و s_2 توسط الگوریتم ۴ توضیح داده شده است. در این الگوریتم $R(t)$ مجموعه کلاس‌هایی است که در دوره‌ی زمانی t اشغال شده‌اند. r_1 کلاس اختصاص داده‌شده به زیرجلسه‌های جلسه‌ی s_1 و r_2 کلاس اختصاص داده‌شده به زیرجلسه‌های جلسه‌ی s_2 است. تعریف نمادهای دیگر را می‌توان در الگوریتم‌های قبلی یافت. با توجه به دو جلسه‌ی s_1 و s_2 که به ترتیب به کلاس‌های r_1 و r_2 مرتبط هستند، ابتدا بررسی می‌کنیم که آیا r_1 به مجموعه کلاس‌های امکان‌پذیر درس s_2 تعلق دارد و بالعکس. اگر جواب منفی باشد، نمی‌توانیم کلاس‌های s_1 و s_2 را باهم عوض کنیم. در غیر این صورت، برای هر دوره‌ی زمانی t از S_t، s_1 را در نظر می‌گیریم. اگر یکی از این دروس در کلاس r_1 برنامه‌ریزی‌شده باشد، نمی‌توانیم کلاس‌های s_1 و s_2 را باهم عوض کنیم. همین رویه را برای دوره‌های زمانی s_2 انجام می‌دهیم و درصورتی‌که متوجه شویم مبادله غیرممکن است، متوقف می‌شویم. اگر به این نتیجه رسیدیم که مبادله امکان‌پذیر است، پس از انجام مبادله، جریمه را محاسبه می‌کنیم و اگر مبادله منجر به کاهش جریمه شود، مبادله را انجام می‌دهیم.

الگوریتم ۴- مبادله‌ی کلاس‌های دو جلسه‌ی s1 و s2.

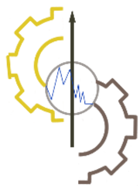
Algorithm 4- exchanging classes of two sessions s1 and s2.

اگر $r_1 \notin PR(C(s_2))$ یا $r_1 \notin PR(C(s_2))$ آنگاه برگردید.
اگر وجود داشته باشد $t \in k(s_1)$ به‌گونه‌ای که $r_1 \in R(t)$ یا وجود داشته باشد $t \in k(s_2)$ به‌گونه‌ای که $r_2 \in R(t)$ آنگاه برگردید.
$P_{old}(s_1,s_2)$ و $P_{new}(s_1,s_2)$ را محاسبه کنید.
اگر $P_{old}(s_1,s_2) < P_{new}(s_1,s_2)$ آنگاه،
کلاس‌های s_1 و s_2 را مبادله کنید.
داده‌های موجود در لیست شاخص را بر این اساس به‌روز کنید.

الگوریتم ۵ را برای کاهش جریمه، با استفاده از چهار همسایگی بکار می‌بریم. در هر تکرار ابتدا دو جلسه را به‌صورت تصادفی انتخاب



صادق شیری و همکاران / مدیریت نوآوری و راهبردهای عملیاتی، دوره ۳، شماره ۴، زمستان ۱۴۰۱، صفحه: ۴۶۴-۴۵۲



کرده و بر روی آن‌ها مبادله‌ی دوره‌های زمانی را اعمال می‌کنیم. سپس یک جلسه‌ی تصادفی s را انتخاب کرده و جایگزینی دوره‌های زمانی را روی آن انجام می‌دهیم. در مرحله‌ی بعد، دو جلسه را به‌طور تصادفی انتخاب کرده و روی آن‌ها مبادله‌ی کلاس‌ها را اعمال می‌کنیم. درنهایت یک جلسه‌ی تصادفی s را انتخاب کرده و جایگزینی کلاس‌ها را روی آن اعمال می‌کنیم. معیار توقف زمان محاسبه مشخص شده است.

الگوریتم ۵- تلاش برای کاهش جریمه.

Algorithm 5- Trying to reduce the fine.

```
مرحله:
    اگر به محدودیت زمانی رسیده است، برگردید.
    دو جلسه‌ی s1 و s2 را به‌طور تصادفی انتخاب کنید.
    الگوریتم ۳ را روی s1 و s2 اعمال کنید.
    به‌طور تصادفی یک جلسه را انتخاب کنید.
    الگوریتم جایگزینی دوره‌های زمانی را برای S اعمال کنید.
    دو جلسه‌ی s1 و s2 را به‌طور تصادفی انتخاب کنید.
    الگوریتم ۴ را روی s1 و s2 اعمال کنید.
    به‌طور تصادفی یک جلسه را انتخاب کنید.
    الگوریتم جایگزینی کلاس را روی S اعمال کنید.
    به مرحله بروید.
```

به‌راحتی می‌توان دید که در مرحله‌ی دوم معادله و جایگزینی دوره‌های زمانی و کلاس‌ها نیاز به $O\left(|Q|^2\right)$ محاسبات دارد؛ بنابراین با توجه به متناهی بودن تعداد تکرارهای الگوریتم مرحله‌ی دوم می‌توان دریافت که محاسبات الگوریتم مرحله‌ی دوم $O\left(|Q|^2\right)$ است.

۵- نتایج عددی

در این بخش، کارایی عددی الگوریتم پیشنهادی را بررسی می‌کنیم. الگوریتم پیشنهادی را با استفاده از زبان برنامه‌نویسی *ANSI C* تحت سیستم‌عامل ۶۴ بیتی ویندوز ۱۰ با پردازنده‌ی مرکزی *Intel corei7* با سرعت ۳/۵ گیگاهرتز و ۸ گیگابایت حافظه پیاده‌سازی کردیم. الگوریتم پیشنهادی را بر روی هفت نمونه ورودی دنیای واقعی *UCTP* اعمال کردیم. ویژگی‌های نمونه‌های ورودی در سه ستون اول **جدول ۱** خلاصه شده‌اند. در **جدول ۱**، *n* نشان‌دهنده‌ی عدد نمونه، *#grs* تعداد گروه‌ها، *#trs* تعداد اساتید، *#lcrs* نشان‌دهنده تعداد زیرجلسات است. برای هر نمونه، الگوریتم پیشنهادی را اعمال کرده و میانگین داده‌های به‌دست‌آمده را در **جدول ۱** ثبت کردیم. در **جدول ۱**، *penI* نشان‌دهنده‌ی جریمه‌ی جواب شدنی به‌دست‌آمده در مرحله‌ی اول است و داده‌های زیرستون‌ها که با ۵ دقیقه، ۱۰ دقیقه و ۱۵ دقیقه نشان داده‌شده‌اند، جریمه‌های کاهش‌یافته پس از ۵، ۱۰ و ۱۵ دقیقه اجرای الگوریتم مرحله‌ی دوم به ترتیب بر روی جواب محاسبه‌شده‌ی مرحله‌ی اول هستند. در اینجا توجه داشته باشید که ازآنجایی‌که مدل الگوریتم ارایه‌شده جدید است، مقایسه با سایر رویکردها امکان‌پذیر نیست.

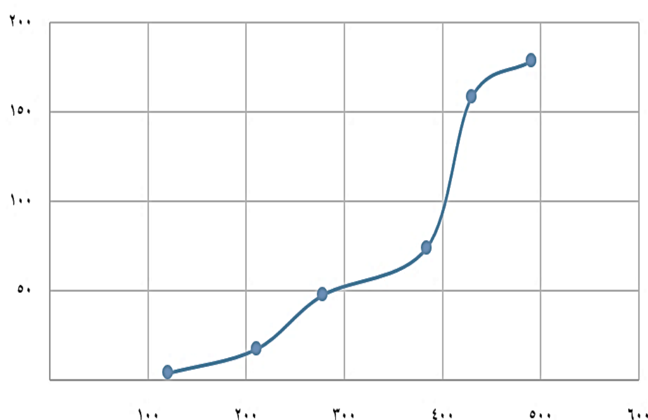
جدول ۱- نتایج عددی مرحله‌ی اول و مرحله‌ی دوم.

Table 1- Numerical results of the first stage and the second stage.

۱۵ دقیقه	۱۰ دقیقه	۵ دقیقه	penI	t	#grs	#trs	#lcrs	n
5403(14%)	5490(12%)	5631(10%)	6243	4.15	5	20	120	1
9771(23%)	9886(23%)	10057(21%)	12730	17.53	9	57	210	2
11232(27%)	11560(25%)	12212(20%)	15306	47.73	11	57	278	3
15303(26%)	15314(26%)	15632(25%)	20588	74.18	13	60	384	4
16539(31%)	16701(30%)	17506(26%)	23833	158.43	15	61	430	5
19435(26%)	19766(25%)	20518(22%)	26340	178.32	16	62	490	6

به‌خوبی شناخته‌شده است که نمونه‌هایی با حدود ۴۰۰ جلسه بزرگ هستند. با توجه به این ورودی، نمونه‌های ۴، ۵ و ۶ بزرگ هستند. نمونه‌های شماره‌های ۱، ۲ و ۳ اندازه‌ی متوسطی دارند. نتایج عددی **جدول ۱** نشان می‌دهد که در همه‌ی موارد، الگوریتم پیشنهادی با موفقیت، جوابی شدنی در مرحله‌ی اول پیداکرده و جریمه را در مرحله‌ی دوم به‌طور موثر کاهش می‌دهد. توجه داشته باشید که فشرده بودن زیرجلسه‌ی دروس و یکنواختی کلاس‌ها را به‌عنوان محدودیت‌های سخت در نظر گرفته‌ایم و در زمان محاسبات معقول (کمتر از ۵ دقیقه) به جوابی دست‌یافته‌ایم که تمامی محدودیت‌های سخت را برآورده کند. برای بررسی کارایی روش ارایه شده برای کاهش جریمه، برای هر

نمونه، مرحله‌ی دوم الگوریتم را بر روی جواب محاسباتی به مدت ۵، ۱۰ و ۱۵ دقیقه اعمال کردیم و جریمه‌ی کاهش‌یافته را در **جدول ۱** ثبت کردیم. نتایج عددی نشان‌دهنده‌ی کارایی الگوریتم ارایه شده در کاهش جریمه است.



شکل ۱- نمودار زمان برحسب تعداد دروس.

Figure 1- Time chart according to the number of courses.

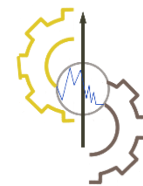
شکل ۱، نمودار تغییر زمان برحسب تعداد دروس را نشان می‌دهد. همان‌طور که از **شکل ۱** پیداست با افزایش تعداد دروس، زمان افزایش می‌یابد. علاوه بر این الگوریتم پیشنهادی توانسته است مسایل بزرگی با ۴۹۰ درس را در مدت‌زمان معقول ۱۷۸/۳ ثانیه حل کند. نرخ رشد زمان حل نسبت به افزایش تعداد دروس وابسته به ساختار مساله است. به‌عنوان‌مثال وقتی تعداد دروس از ۳۸۴ به ۴۳۰ افزایش پیدا می‌یابد زمان حل به‌اندازه‌ی ۸۴/۲۵ ثانیه افزایش داشته است، درحالی‌که وقتی تعداد دروس از ۴۳۰ به ۴۹۰ درس افزایش داشته است، زمان حل تنها به‌اندازه‌ی ۱۹/۸ ثانیه افزایش داشته است.



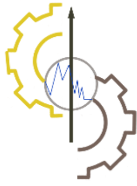
شکل ۲- مقایسه‌ی مقدار و درصد کاهش جریمه نسبت به تعداد دروس.

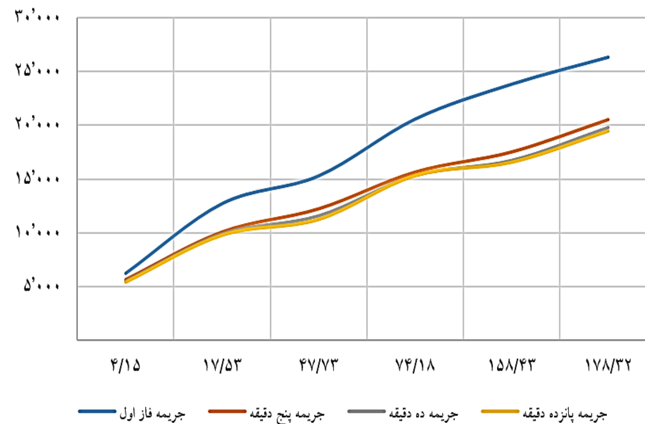
Figure 2- Comparison of the amount and percentage of reduction of penalty in relation to the number of courses.

در **شکل ۲**، مقدار جریمه‌ی جواب‌های به‌دست‌آمده و کاهش ایجادشده در جریمه‌ها نسبت به افزایش تعداد دروس موردبررسی قرارگرفته است. همان‌گونه که در نمودار آمده است با افزایش تعداد دروس، مقدار جریمه افزایش یافته است. در همه‌ی موارد با اعمال مرحله‌ی دوم، مقدار جریمه‌ی جواب شدنی یافت شده به‌طور قابل‌ملاحظه‌ای کاهش یافته است. در همه‌ی موارد، در پنج دقیقه‌ی اول، مقدار جریمه به‌طور متوسط به‌اندازه‌ی ٪۲۰ کاهش یافته است. ولی مقدار کاهش در پنج دقیقه‌ی دوم به‌طور متوسط به‌اندازه‌ی ٪۳ و در پنج دقیقه‌ی سوم کاهش جریمه به‌طور متوسط به‌اندازه‌ی ٪۱ بوده است؛ بنابراین نتایج عددی به‌نوعی نشان می‌دهند که با اعمال مرحله‌ی دوم روش، پس از مدت‌زمان اندکی، جریمه‌ی جواب شدنی کم شده و با احتمال زیاد جواب‌های به‌دست‌آمده به جواب‌های بهینه نزدیک هستند.



صادق شیری و همکاران / مدیریت نوآوری و راهبردهای عملیاتی، دوره ۳، شماره ۴، زمستان ۱۴۰۱، صفحه: ۴۶۴-۴۵۲





شکل ۳- نسبت کاهش جریمه به زمان حل مساله.

Figure 3- Ratio of penalty reduction to problem solving time.

شکل ۳، نشان می‌دهد که با افزایش زمان حل و بزرگ شدن مساله، کارایی الگوریتم ارایه شده در مرحله‌ی دوم به‌منظور بهبود جواب شدنی یافت شده، بیشتر می‌گردد. برای مساله‌ای که زمان حل آن ۱۷۸/۳۲ بوده است درصد کاهش پس از ۱۵ دقیقه ۲۶% بوده است درحالی‌که این درصد در مورد مساله بازمان حل ۴/۱۵ برابر ۱۴% است. این بدان معنی است که پس‌ازآن که الگوریتم مربوط به مرحله‌ی اول توانست یک جواب شدنی برای مساله را در زمان معقول به دست آورد، الگوریتم مربوط به مرحله‌ی دوم می‌تواند مطلوبیت برنامه برای مدیریت و اساتید مربوطه را به‌طور قابل‌توجهی افزایش دهد. علاوه بر این، همان‌طور که قبلا بحث شده کاهش جریمه بسیار کند در دقایق بالای پنج دقیقه ایجاب می‌کند که جواب حاصل با احتمال زیاد به جواب بهینه نزدیک است.

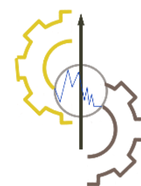
در پایان شایان‌ذکر است که روش ارایه شده در این مقاله با روش‌های ارایه شده در سایر مقالات مرتبط، از نمایش جواب جدید و روندهایی متناسب با نمایش جواب جدید استفاده می‌کند. علاوه بر این، همان‌طور که قبلا ذکر شد یکی دیگر از ویژگی‌هایی که الگوریتم ارایه شده از سایر الگوریتم‌ها متمایز می‌کند دومرحله‌ای بودن آن است. به‌بیان‌دیگر، سایر الگوریتم‌های فرا ابتکاری ارایه شده معمولا همزمان با برآورده شدن محدودیت‌های سخت، سعی در مینیمم‌سازی جریمه‌ی جواب‌های شدنی می‌کند. آنچه مسلم است این است که با توجه به نتایج عددی ارایه شده توسط آولا و واسیلو (۲۰۰۵)، اندازه‌ی مسایلی که در این مقاله حل می‌شوند بسیار بزرگ‌تر از مسایلی هستند که توسط روشی دقیق حل می شوند. نکته‌ی دیگری که در اینجا قابل‌ذکر است این است که در اکثر مقالات دیگر که توسط روش‌های فرا ابتکاری یا ابتکاری حل‌شده‌اند حداقل یک یا چند محدودیت بسیار متفاوت از محدودیت‌هایی بوده‌اند که در اینجا موردبررسی قرار گرفته‌اند ولی به‌هرحال می‌توان گفت که اندازه‌ی مسایل و زمان حل آن‌ها توسط روش ارایه شده در این مقاله قابل‌مقایسه با کمیت‌های متناظر در مقالات مرتبط هست.

۶- بحث و نتیجه‌گیری

در این مقاله، یک روش ابتکاری جدید و کارآمد برای حل یک مدل خاص از زمان‌بندی دروس دانشگاهی ارایه کردیم. این مدل دارای محدودیت‌های سخت جدید مانند فشرده بودن زیرجلسه‌های یک درس و سازگاری کلاس‌های آن‌ها است. الگوریتم پیشنهادشده روشی را برای حل جدول زمانی دروس دانشگاهی بدون استفاده از هیچ الگوریتم تطبیقی ارایه می‌دهد. همچنین روش ارایه شده از نقض قیود سخت در مرحله‌ی اول و جریمه‌ی جواب در مرحله‌ی دوم استفاده می‌کند. درنهایت، روش ارایه شده را بر روی برخی از نمونه‌های بزرگ دنیای واقعی اعمال کردیم. نتایج عددی نشان‌دهنده کارایی الگوریتم پیشنهادی است.

به‌عنوان پیشنهاد برای ادامه‌ی پژوهش در این زمینه، می‌توان از ایده‌ی سایر روش‌های فراابتکاری مانند تبرید شبیه‌سازی‌شده، کلونی مورچگان و غیره در الگوریتم‌های مرحله‌ی اول و مرحله‌ی دوم استفاده کرد. علاوه بر این می‌توان تلاش کرد تا در صورت امکان یک مدل برنامه‌ریزی صحیح برای مساله ارایه کرد و کارایی آن را با روش ارایه شده مقایسه نمود. یکی دیگر از جنبه‌هایی که می‌تواند موردبررسی قرار گیرد، در نظر گرفتن سایر معیارهای مطلوبیت در الگوریتم‌های پیشنهادی است. به‌عنوان‌مثال می‌توان تلاش کرد که همه‌ی کلاس‌های یک گروه از دانشجویان در یک روز برگزار شود. یک معیار دیگر که می‌تواند در نظر گرفته شود تلاش به‌منظور مینیمم کردن تعداد کلاس‌ها و منابع دیگر باشد.

ازجمله محدودیت‌های مدل و روش پیشنهادی این است که کاملا برای مساله‌ی زمان‌بندی دروس دانشگاهی مبتنی بر برنامه‌ی درس طراحی‌شده است و بنابراین نمی‌توان از آن برای حل مسایل زمان‌بندی دروس دانشگاهی مبتنی بر ثبت‌نام استفاده کرد. یکی دیگر از محدودیت‌های مدل و روش پیشنهادی این است که به دلیل جدید بودن نوع نمایش جواب و اینکه روش ارایه شده کاملا بر اساس استفاده از شکل نمایش جواب است، شاید نتوان از آن برای حل سایر مسایل زمان‌بندی دروس دانشگاهی که از نمایش جواب‌های متفاوتی استفاده می‌کنند، استفاده کرد.



منابع

Avella, P., & Vasil'Ev, I. (2005). A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, *8*(6), 497-514. https://doi.org/10.1007/s10951-005-4780-1

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2022). Educational timetabling: problems, benchmarks, and state-of-the-art results. *European journal of operational research*. 0377-2217. https://doi.org/10.1016/j.ejor.2022.07.011

Chen, R. M., & Shih, H. F. (2013). Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, *6*(2), 227-244. https://doi.org/10.3390/a6020227

Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European journal of operational research*, *160*(1), 106-120. https://doi.org/10.1016/j.ejor.2003.06.023

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European journal of operational research*, *153*(1), 117-135. https://doi.org/10.1016/S0377-2217(03)00103-6

Gaspero, L. D., & Schaerf, A. (2002). Multi-neighbourhood local search with application to course timetabling. *International conference on the practice and theory of automated timetabling*. (pp. 262-275). Springer, Berlin. https://doi.org/10.1007/978-3-540-45157-0_17

Lemos, A., Monteiro, P. T., & Lynce, I. (2021). Disruptions in timetables: a case study at universidade de Lisboa. *Journal of scheduling*, *24*(1), 35-48. https://doi.org/10.1007/s10951-020-00666-3

Martin, C. H. (2004). Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, *34*(6), 460-465. https://doi.org/10.1287/inte.1040.0106

Nguyen, K., Nguyen, Q., Tran, H., Nguyen, P., & Tran, N. (2011). Variable neighborhood search for a real-world curriculum-based university timetabling problem. *Third international conference on knowledge and systems engineering* (pp. 157-162). IEEE. https://doi.org/10.1109/KSE.2011.31

Qualizza, A., & Serafini, P. (2004). *A column generation scheme for faculty timetabling. International conference on the practice and theory of automated timetabling*, (pp. 161-173). Springer. https://doi.org/10.1007/11593577_10

Schimmelpfeng, K., & Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *Or Spectrum*, *29*(4), 783-803. https://doi.org/10.1007/s00291-006-0074-z

Suyanto, S. (2010, June). An informed genetic algorithm for university course and student timetabling problems. *Proceedings of the 10th international conference on Artifical intelligence and soft computing: Part II* (pp. 229-236). https://doi.org/10.1007/s00291-006-0074-z